XC83x

AP08128

inTouch Application Kit - Touch Wheel

Application Note

V1.0, 2012-02

Microcontrollers

Edition 2012-02

Published by
Infineon Technologies AG
81726 Munich, Germany

XC82x/XC83x
Revision History: V1.0 2012-02

Previous Version(s):

Page	Subjects (major changes since last revision)
–	

We Listen to Your Comments
Is there any information in this document that you feel is wrong, unclear or missing? Your feedback will help us to continuously improve the quality of this document. Please send your proposal (including a reference to this document) to:
mcdocu.comments@infineon.com

Table of Contents

1 Introduction

In today's Human-Machine Interface (HMI) designs, capacitive touch technology is now often more widely used than traditional mechanical buttons. Capacitive touch technology is the more popular choice because it brings flexibility, a high-level of customization, and a significant reduction in overall system cost.

The *inTouch Application Kit* is available to help learn about working with the advanced touch solutions provided by Infineon. Step-by-step tutorials covers the basics of Infineon's touch solutions, while example application code can be used to start developing new touch-related projects.

The *inTouch Application Kit* comprises of a mother board, supplied as a USB stick, and a number of daughter boards. **Figure 1** shows the USB stick with the Wheel daughter board.

Among the many different touch input elements that can be designed with capacitive touch technology, the touch wheel is gaining popularity because of the intuitive control it provides. This application note describing the Wheel daughter board, aims to highlight the ease of implementing a design with Infineon's touch solutions. Topics covered include program flow and touch behavior.

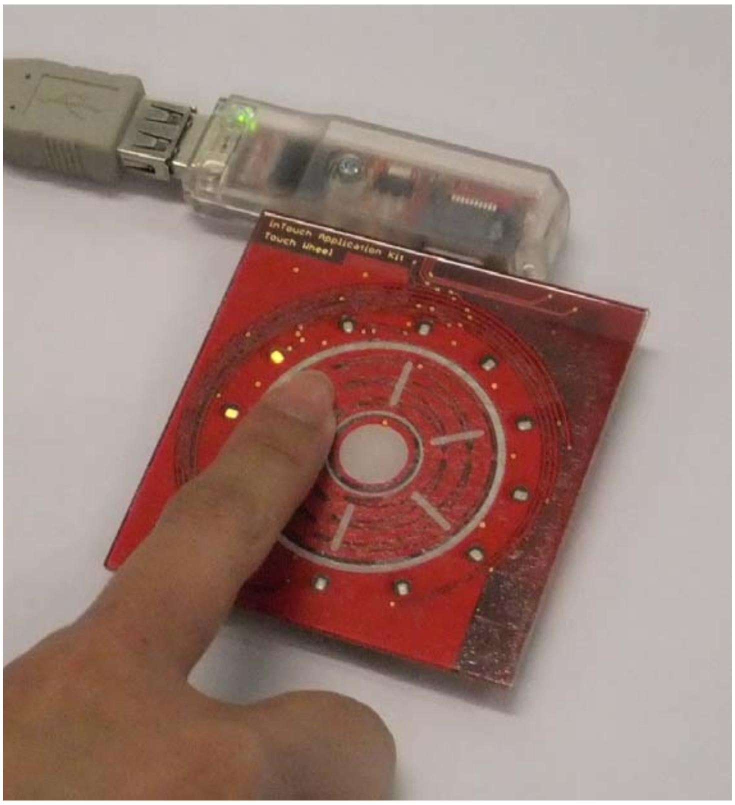

Figure 1 *inTouch Application Kit* (USB Stick and Wheel board)

2 Hardware & Program Flow

This section describes the hardware used and the connections involved.

2.1 Hardware

Infineon's XC836MT 2FRI (**Figure 2**) is used in this application. The XC836MT is embedded in the *inTouch Application Kit*'s USB stick. For more details regarding the USB stick, please refer to *AP08126: Infineon Touch Solutions - inTouch Application Kit*.

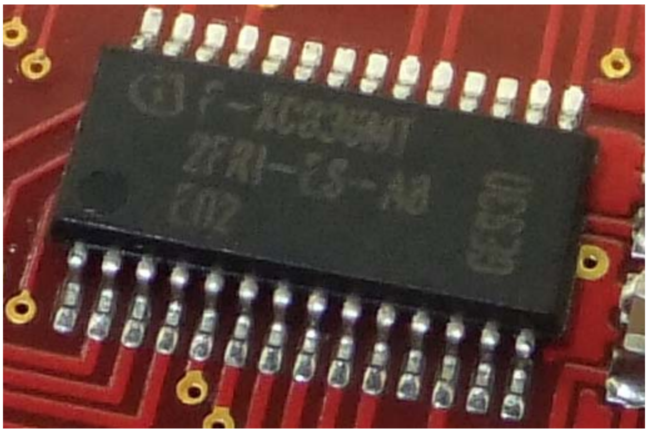

Figure 2 Infineon's XC836MT 2FRI

The *inTouch Wheel* board (**Figure 3**) is available as a plug-in daughter board which is part of the *inTouch Application Kit.*

Figure 3 Wheel Board

The *inTouch Wheel* board is a standard PCB with a 1mm thick plexiglas cover glued to the board. The touch wheel is connected to 3 LEDTS pad inputs of the XC836. The centre button is connected to an LEDTS pad input of the XC836. 12 indicator LEDs share 3 LEDTS column pins and 4 line pins of the XC836. The schematics are available in the **Appendix - Schematics and Layout**.

Users can tap or dial the touch wheel and they can tap the centre button.

2.2 Program Flow

The *inTouch Wheel* board has four touch pads; one is used as a touch button and the remaining three form a wheel for dialling. All 4 pads are handled by the XC836MT microcontroller's LED Touch-Sense Control Unit (*LEDTSCU*) which is a dedicated touch sense controller module. The method for measuring the pad capacitance is the Relaxation Oscillator (RO) Topology.

For more information on the RO Topology, refer to the application note *AP08126: Infineon Touch Solutions - inTouch Application Kit*.

In terms of interrupts, the Time Frame interrupt has the highest priority. In this service routine, touch sense related tasks are performed each time pad capacitance has been measured. LED updates, which are performed in the Time Slice interrupt, have low priority. The Timer 2 (T2) Overflow interrupt is given low priority due to its slow frequency. The UART interrupt has low priority as it is not time-critical. **Figure 4** provides an illustration of the program overview.

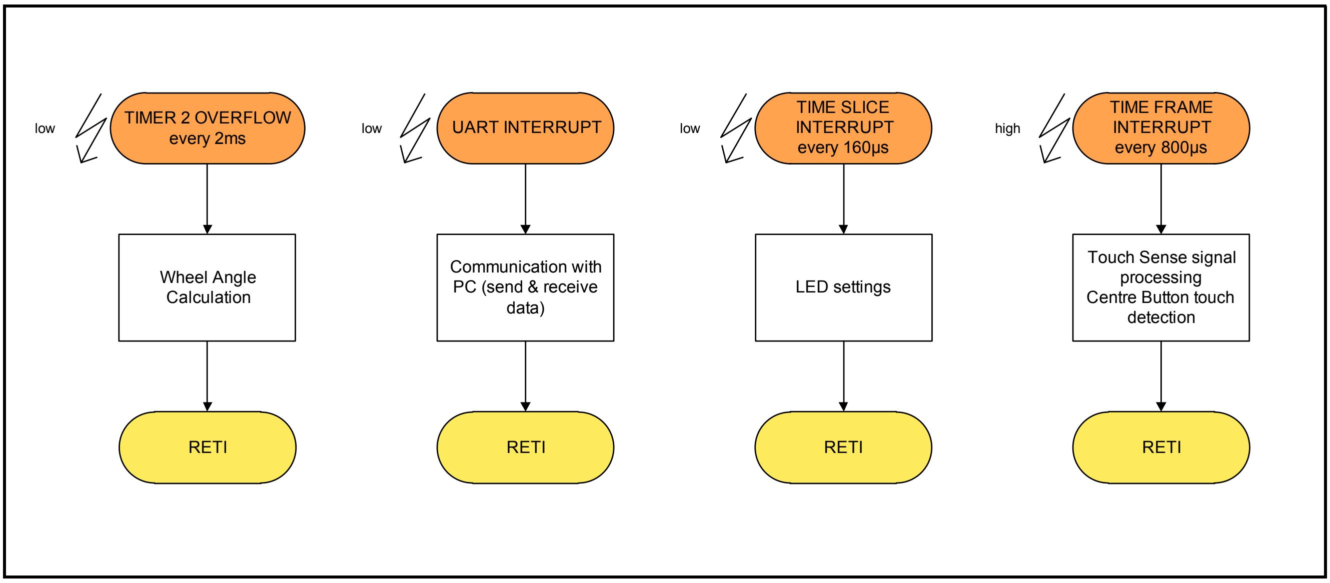

Figure 4 Program Overview

The tasks performed in each interrupt service routine are further illustrated in the flowcharts that follow:

- T2 Overflow Interrupt (**Figure 5**)
  - The T2 module provides a slow time base by generating the T2 Overflow interrupt for calculations necessary to handle the touch wheel.
- UART Interrupt (**Figure 6**)
  - The UART module, which is part of the XC800 core, is used for full-duplex UART communication with the PC.
- Time Slice Interrupt (**Figure 7**)
  - The LEDTSCU module generates this interrupt after every LED column activation where the pattern for the next LED column is loaded into shadow registers.
- Time Frame Interrupt (**Figure 8**)
  - The LEDTSCU module generates this interrupt after every measurement where signal processing and touch detection take place.

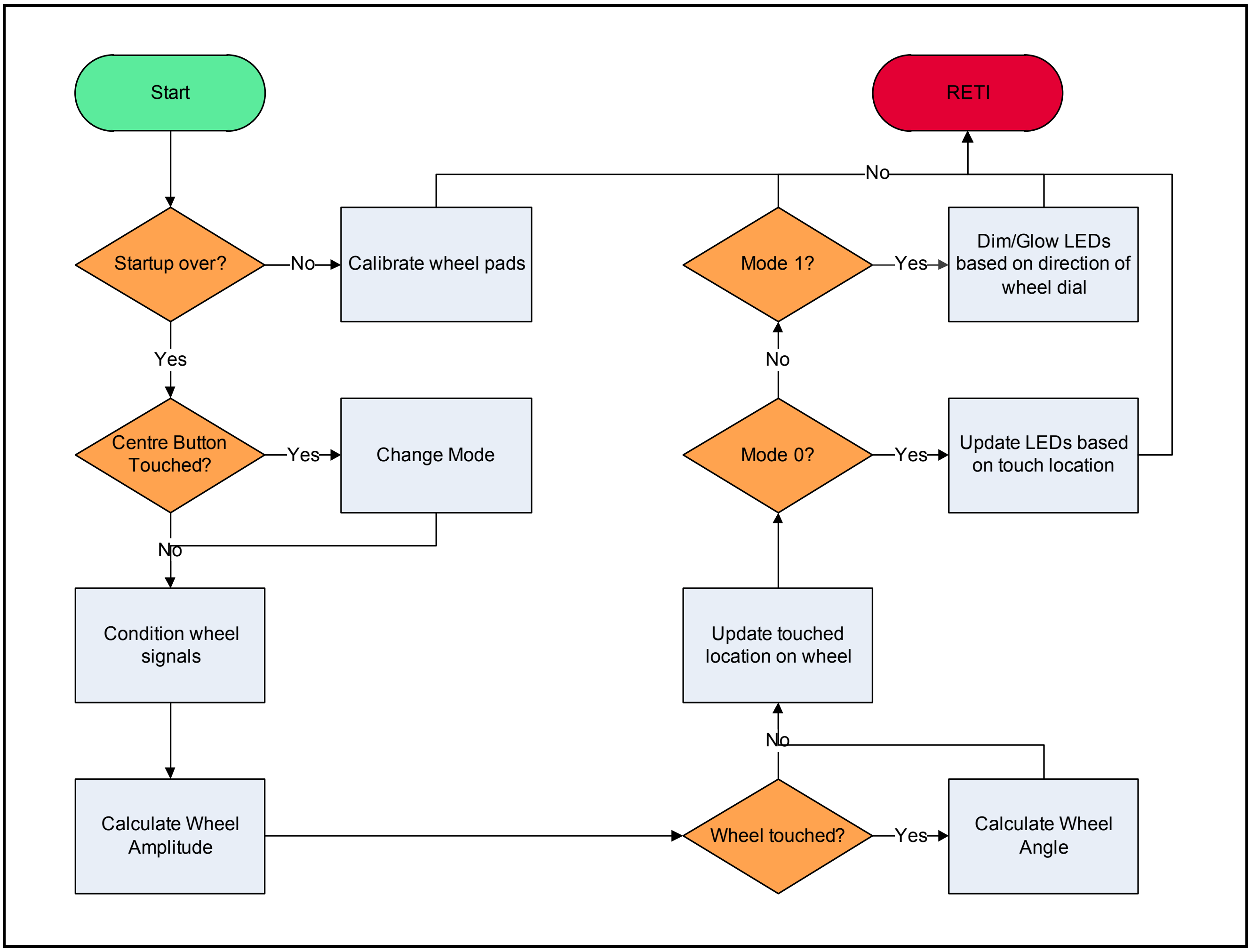

Figure 5 Timer 2 Overflow Interrupt Service Routine

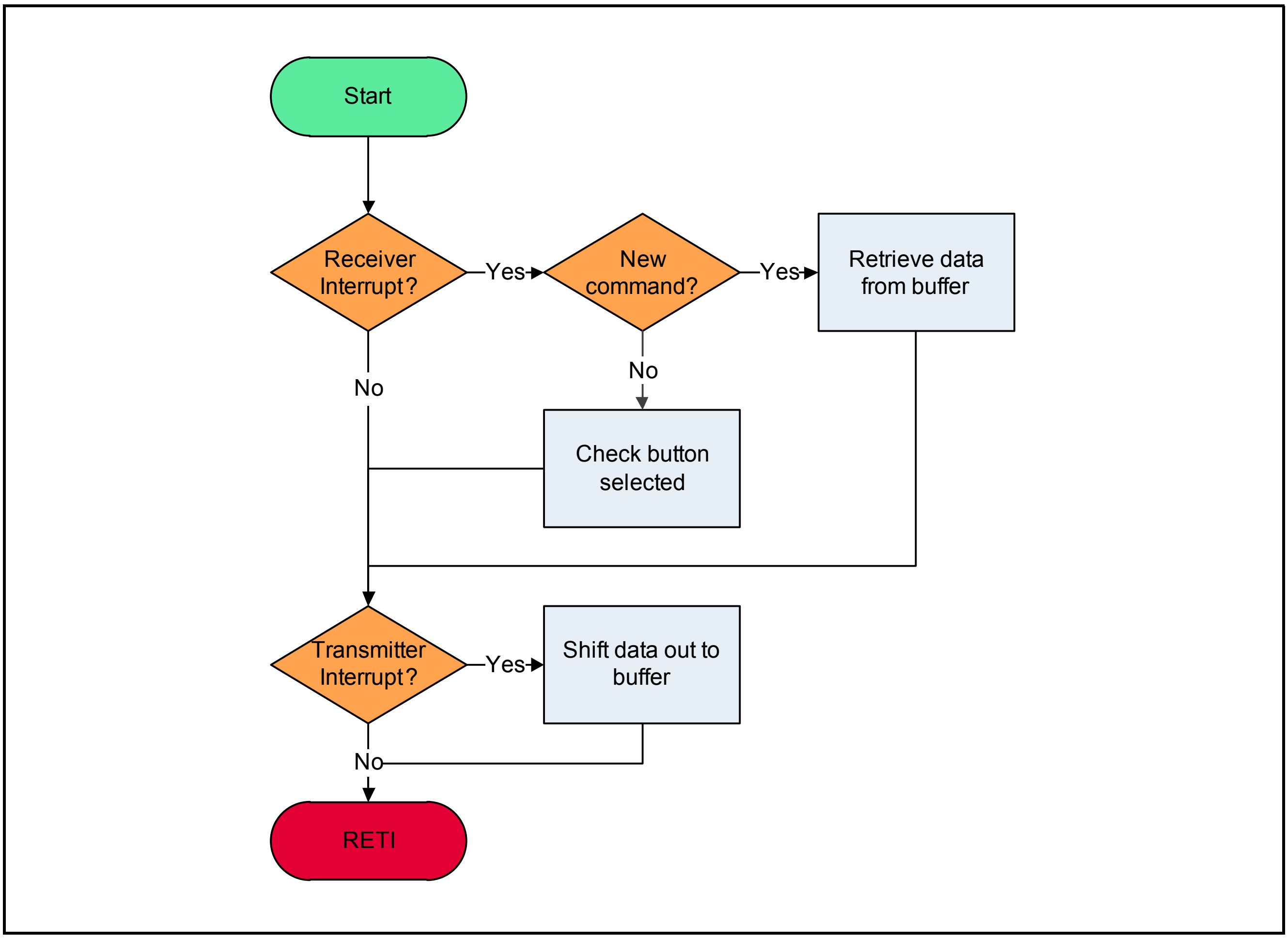

Figure 6 UART Interrupt Service Routine

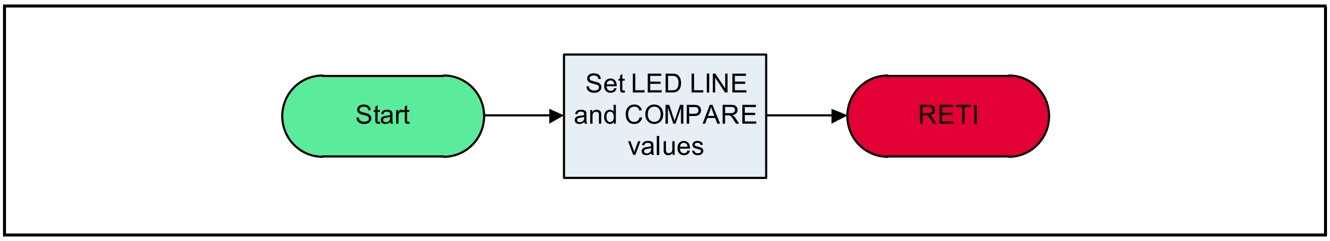

Figure 7 Time Slice Interrupt Service Routine

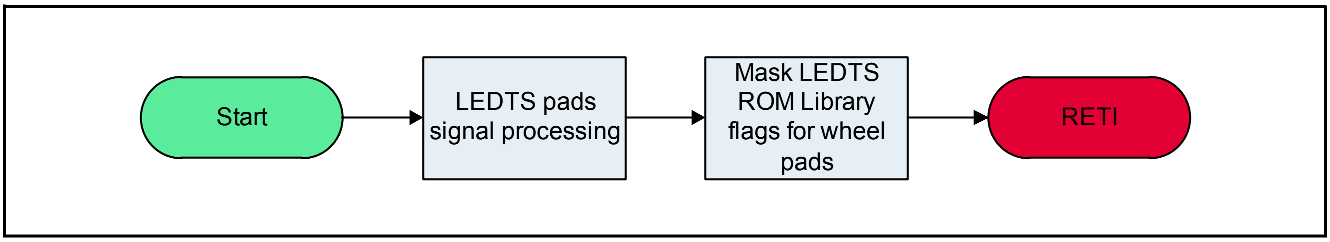

Figure 8 Time Frame Interrupt Service Routine

3 Sensing Touch on Wheel

This section describes how the LEDTSCU module of the XC836, complemented with a software library, control the touch wheel. The algorithm for calculating the location of touch is also explained in the following section.

The main touch sensing functions, handled by software, are as follows:

- Sample accumulation (ROM library)
- Signal filtering and moving average calculation (ROM library)
- Touch detection (ROM library)
- Touch wheel calibration (user software in Flash)
- Signal tuning (user software in Flash)

If properly configured, the LEDTSCU automatically measures the capacitance of the four pads. This capacitance increases when a button is touched. A library function in ROM processes the capacitance signals and detects touch on the centre button. It does so by accumulating 3 samples and low-pass filtering them to obtain a moving average. The moving average filters noise and is used as a reference to detect sudden changes in capacitance. When the button is touched or released, a corresponding pad flag in RAM will be set or reset. For more information on the LEDTS ROM Library, please refer to the XC836 User's Manual.

The pad flags for the wheel pads are unused (always cleared) and it is the moving averages ("*pad averages*") that are used instead to calculate the angle of the touch. The three pads are automatically calibrated to the same sensitivity and resolution during startup. Once the pad averages are stable, an angle calculation algorithm is run if the wheel is touched. The calculated angle is then used to determine the location of touch, and hence the LED to be switched on.

3.1 Wheel Angle Calculation

The three touch pads are placed in a spatially interpolated manner (**Figure 9**).

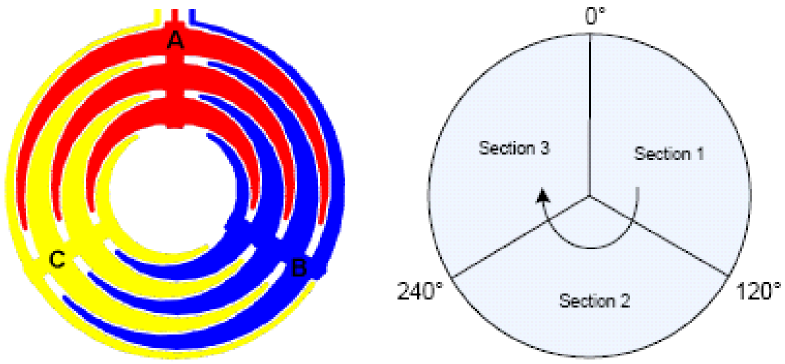

Figure 9 Spatially interpolated wheel layout and abstraction

If the pads are calibrated to roughly the same sensitivity and the wheel is dialed clockwise with constant angular speed and constant pressure (constant effective finger area), the pad average signals are expected to behave in a linear manner in this model as seen in **Figure 10**.

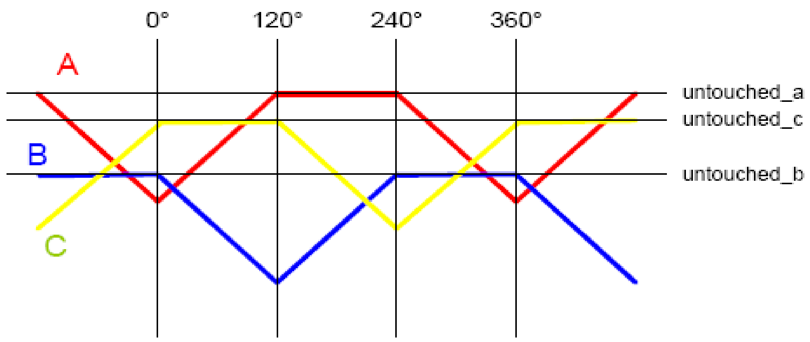

Figure 10 Pad average signals of the three wheel pads during dialling

Values untouched_a, untouched_b and untouched_c are the pad average levels for pads A, B and C respectively when they are not touched.

If the pads have roughly the same sensitivity, the three signals can be tuned to have a common untouched level (**Figure 11**). The actual signals can be expected to look like those in **Figure 12**.

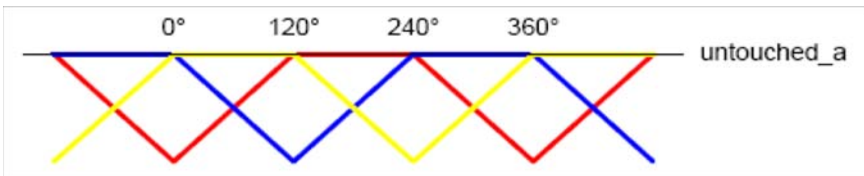

Figure 11 Pad average signals of the three wheel pads after tuning

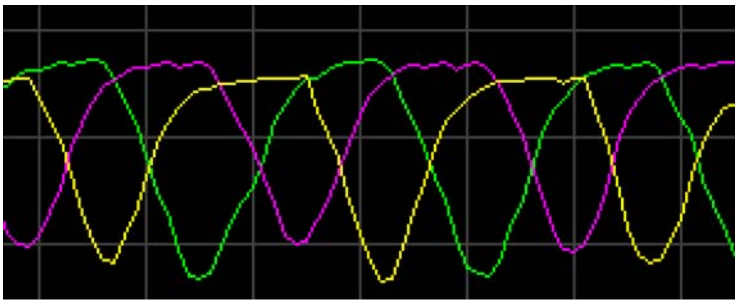

Figure 12 Actual pad average signals after tuning

The, now common, untouched level is very high compared to the difference between touched and untouched states. To make calculations easier, the signals are transformed near to zero by linear combinations which can be represented by the formulae below. **Figure 13** provides an illustration of the transformation. This transformation also makes the transitions between angle sections smooth, which is especially important if the three pads have different sensitivity or unstable untouched levels due to imperfect calibration or a changing environment.

$$X = \frac{A+B}{2} - C \qquad Y = \frac{A+C}{2} - B \qquad Z = \frac{B+C}{2} - A$$

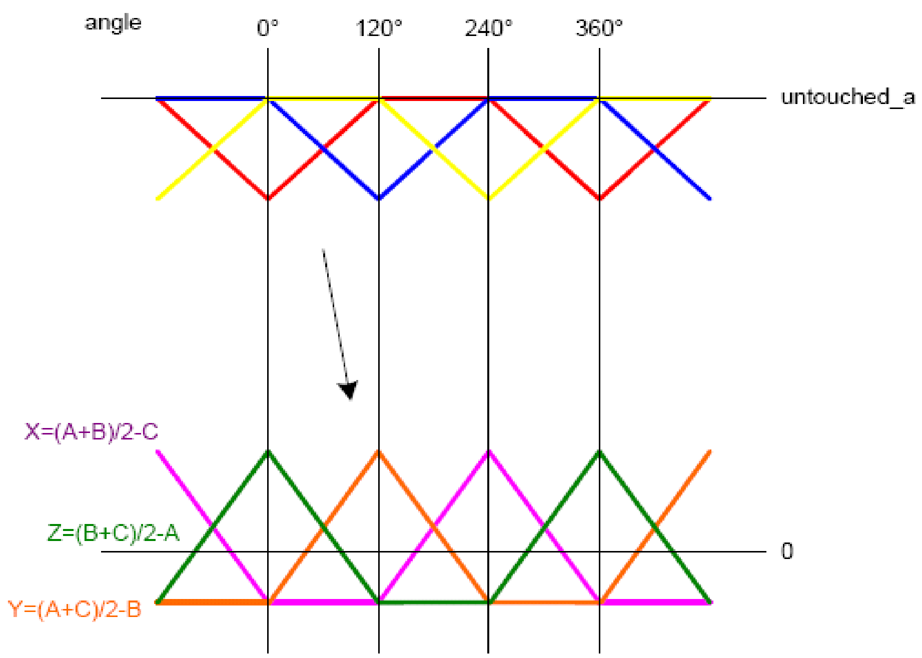

Figure 13 Combined pad average signals

The resulting *X*, *Y* and *Z* signals still have three distinct sections between 0° to 360°.

Section 1 (0° to 120°)

Before the transformation, Section 1 has three signals between UT and UT-MAXT (**Figure 14**). UT stands for the untouched level and UT-MAXT stands for the signal level when the largest area of the respective pad is touched (this happens at 0°, 120° and 240°).

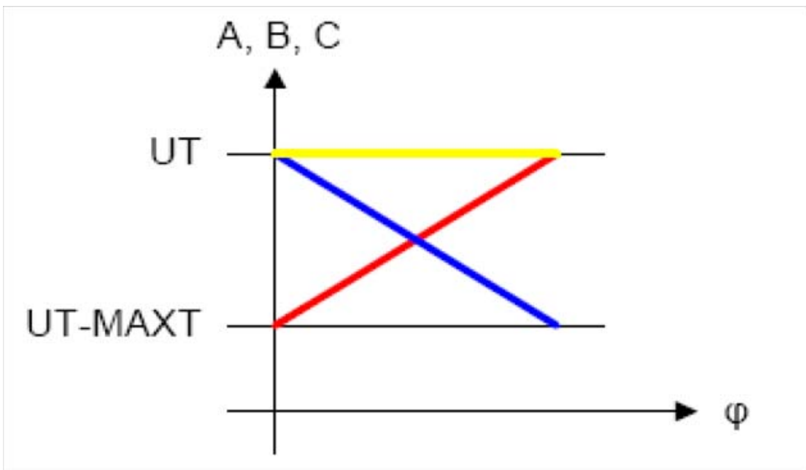

Figure 14 Section 1 before transformation

After the transformation, the *X*, *Y* and *Z* signals have much lower values (**Figure 15**). The angle axis has been arbitrarily scaled from -1 to 2 in this region for convenience.

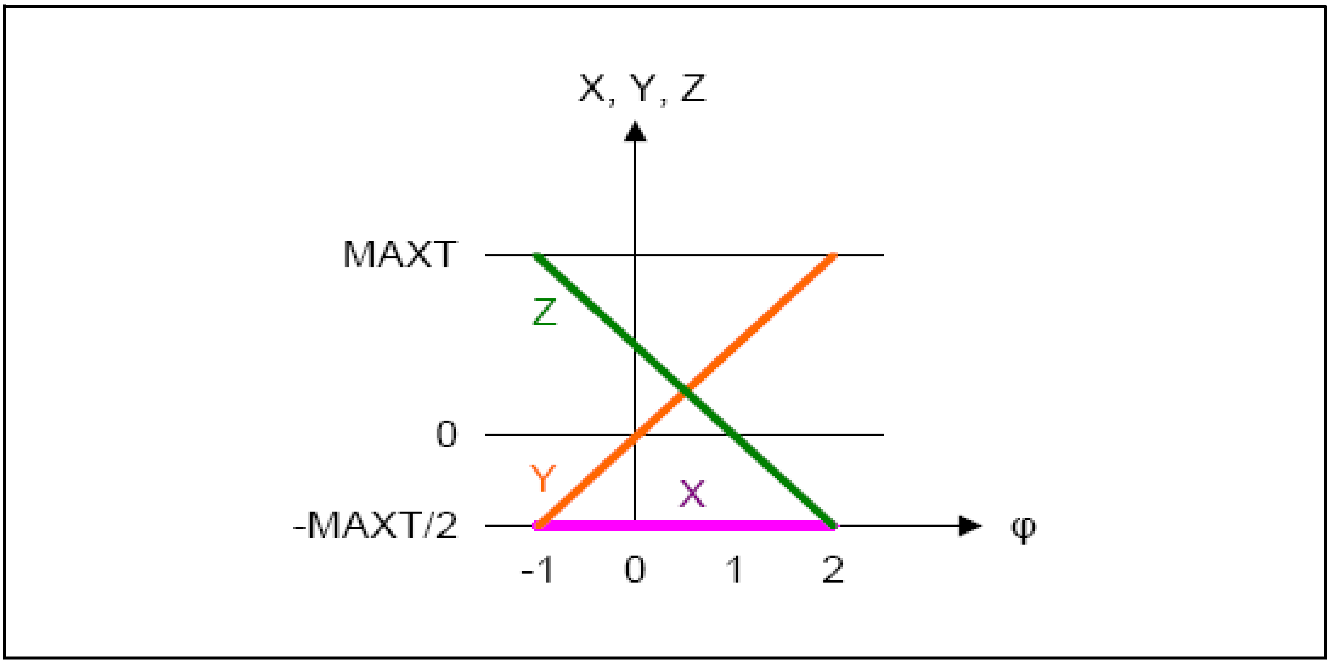

Figure 15 Section 1 after transformation

Signal *X* is constant low in this section so it does not participate in the angle calculation. The other two signals can be described as:

$$Y = \frac{MAXT}{2}\varphi \tag{1}$$

$$Z = \frac{MAXT}{2} - \frac{MAXT}{2}\varphi \tag{2}$$

If we rearrange **Equation (1)**, we get $\frac{MAXT}{2} = \frac{Y}{\varphi}$ which we can substitute in **Equation (2)**:

$$Z = \frac{Y}{\varphi} - \frac{Y}{\varphi}\varphi$$

$$Z = \frac{Y}{\varphi}(1-\varphi)$$

$$\varphi(Y+Z) = Y$$

$$\varphi = \frac{Y}{Y+Z} \tag{3}$$

$$\varphi = 1 - \frac{Z}{Y+Z} \tag{4}$$

One division is needed to calculate the angle; this operation needs the most computing performance. To minimize the error, it is safer to use **Equation (3)** if *Y* is larger and **Equation (4)** if *Z* is larger.

An offset of 1 and a scaling factor of 2^R are added to create a more usable calculated angle (**Figure 16**). R is for resolution and corresponds to the number of left bitshifts on the numerator.

Section 1 Left $$\varphi = 2 \times 2^R - \frac{Z \times 2^R}{Y + Z}$$

Section 1 Right $$\varphi = \frac{Y \times 2^R}{Y + Z} + 2^R$$

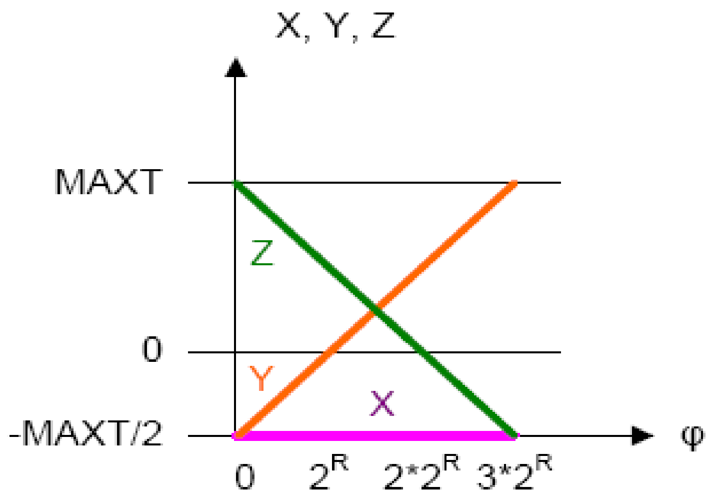

Figure 16 Section 1 after offsetting and scaling

Sections 2 and 3

In these two sections, the angle can be calculated in a similar way as described in **Section 1**, using the two non-constant signals. Offsets of 4 and 7, and the same scaling factor, can then be added to sections $120^\circ - 240^\circ$ and $240^\circ - 360^\circ$ respectively to get a calculated angle of 0..9*2R for $0^\circ..360^\circ$.

Section 2 Left $$\varphi = 5 \times 2^R - \frac{Y \times 2^R}{X + Y}$$

Section 2 Right $$\varphi = \frac{X \times 2^R}{X + Y} + 4 \times 2^R$$

Section 3 Left $$\varphi = 8 \times 2^R - \frac{X \times 2^R}{X + Z}$$

Section 3 Right $$\varphi = \frac{Z \times 2^R}{X + Z} + 7 \times 2^R$$

Figure 17 gives an illustration of the calculated angle across all 3 sections.

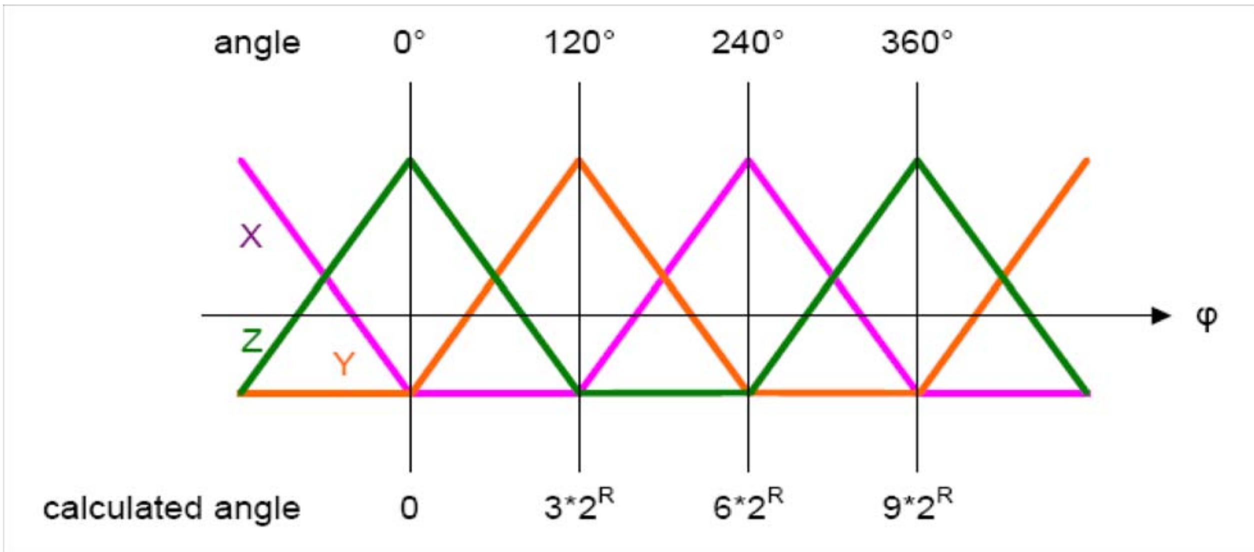

Figure 17 Calculated angle vs real angle across all sections

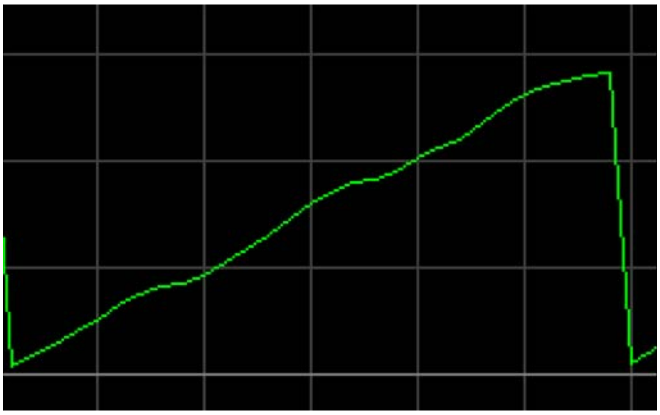

Figure 18 Actual calculated angle for a full round

Infineon provides a function library for angle calculation. The *resolution*, explained earlier, is user selectable from 1 to 8. The XC836M and XC836MT microcontrollers have a Multiplication/Division Unit (MDU) for hardware acceleration. If the MDU is used for the division necessary to calculate the angle, the *resolution* is fixed at 8, execution is faster and the code size is about 250 bytes smaller than without hardware acceleration. The disadvantage is that the MDU increases the microcontroller's current consumption by almost 1mA.

4 U-SPY

For the *inTouch Wheel* board, one settings file, **inTouch_Wheel.ini** has been configured.

4.1 inTouch_Wheel.ini

This settings file (**Figure 19**) is customized to allow the user to monitor the calculated wheel angle, the brightness level of the LEDs, the parameters of LEDTS ROM library and the Touch Wheel Library while running the demonstration program.

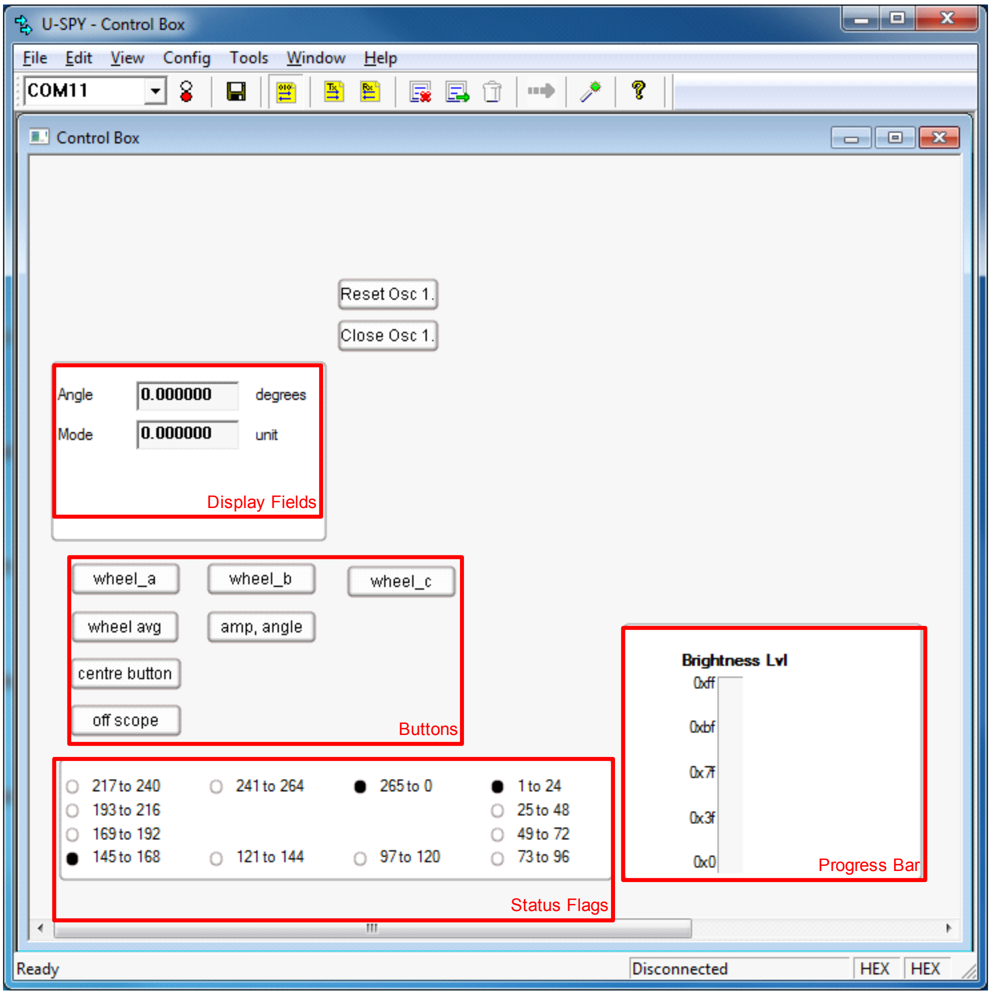

Figure 19 inTouch_Wheel.ini User Interface

Buttons

In this settings file, the buttons allow the user to choose the signal(s) which they would like to monitor. The format of the data transmitted for the buttons is in the following format (**Table 1**):

Table 1 Transmit Data Format for Buttons

	D0	D1
Value (hex)	08	XX
Description	I.D. number	Button number

The data received by the microcontroller will be used to determine the signals that will be transmitted to U-SPY for display either as **Status Flags** (default) or on the **Oscilloscope**. The "off scope" button provides the option for the user to return to the default display mode (status flags) after monitoring the signals on the oscilloscope.

Status Flags

The format of the transmitted data for the status flags is as follows (**Table 2**):

Table 2 Transmit Data Format for Status Flags

	D0	D1	D2
Value (hex)	A3	XX	XX
Description	I.D. number	Mask (High Byte)	Mask (Low Byte)

The statuses of the LEDs received by USpy are masked before they are displayed as status flags. It is important that the bits of a mask do not overlap with the bits of another mask. This is to ensure that status flags are not falsely turned on. The masks used are as follows (**Table 3**):

Table 3 LED masks for Status Flags

Flag Index	**8**	**12**	**13**	**14**	**15**	**11**
Mask (hex)	0002	0004	0008	0010	0020	0040
Wheel Angle (degrees)	265 to 0	1 to 24	25 to 48	49 to 72	73 to 96	97 to 120
Flag Index	**7**	**3**	**2**	**1**	**0**	**4**
Mask (hex)	0080	0100	0200	0400	0800	1000
Wheel Angle (degrees)	121 to 144	145 to 168	169 to 192	193 to 216	217 to 240	241 to 264

Display Fields

The display fields output the calculated wheel angle and the current active mode. The mode can be toggled by tapping on the touch wheel's centre button. The format of the transmitted data for the display field is as follows (**Table 4**):

Table 4 Transmit Data Format for Display Field

	D0	D1	D2	D3
Value (hex)	A1	XX	XX	XX
Description	I.D. number	Display Field Index	Angle or Mode (High Byte)	Angle or Mode (Low Byte)

Progress Bar

The progress bar only becomes active in Mode 1 (Brightness Control Mode). The mode can be toggled by tapping on the touch wheel's centre button. The format of the transmitted data for the progress bar is as follows (**Table 5**):

Table 5 Transmit Data Format for Progress Bar

	D0	D1	D2	D3
Value (hex)	A2	XX	XX	XX
Description	I.D. number	Progress Bar Index	Brightness Level (High Byte)	Brightness Level (Low Byte)

Oscilloscope

The oscilloscope function allows the user to monitor up to 3 signals at a time (**Figure 20**). A total of 3 oscilloscopes are available. However, we will display only 3 signals on 1 oscilloscope in this application. The format of the transmitted data for the oscilloscope is as follows (**Table 6**):

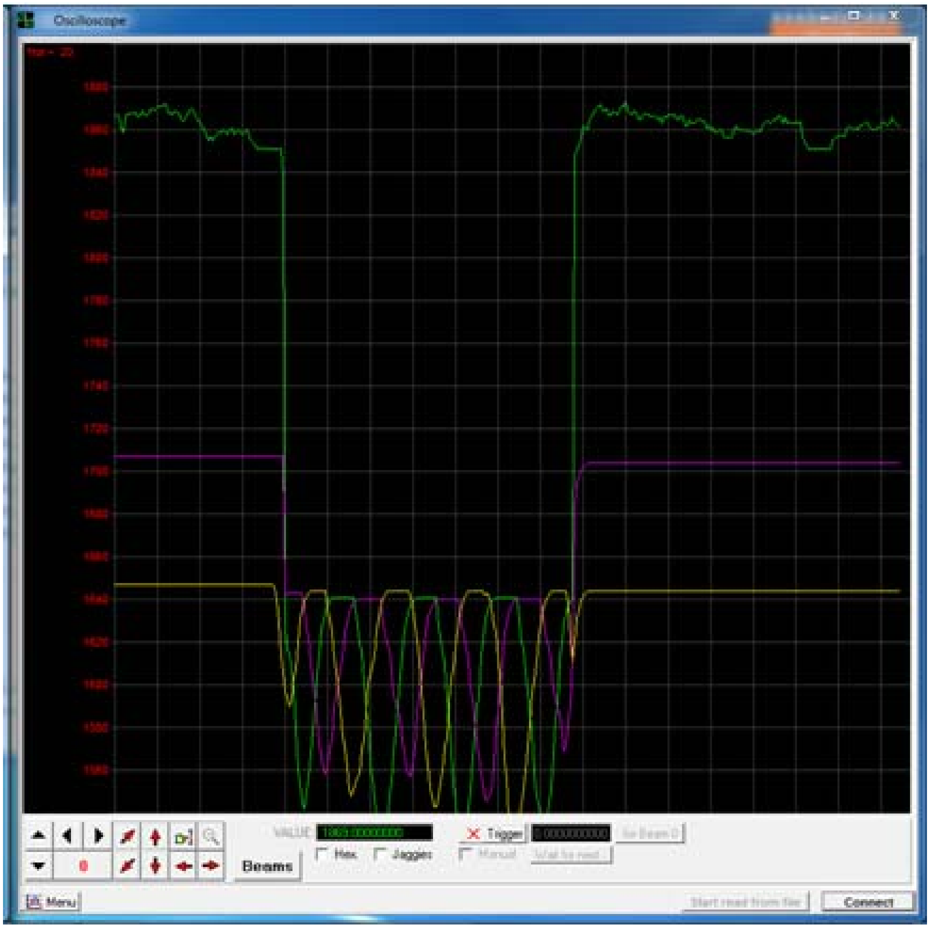

Figure 20 U-SPY Oscilloscope

Table 6 Transmit Data Format for Oscilloscope

	D0	D1	D2	D3	D4	D5	D6	D7
Value (hex)	A4	01	XX	XX	XX	XX	XX	XX
Description	I.D. number	Scope number	Signal 1 high byte	Signal 1 low byte	Signal 2 high byte	Signal 2 low byte	Signal 3 high byte	Signal 3 low byte

As mentioned in the previous section, the user is able to monitor six different types of signals in this settings file. The signals displayed are as follows (**Table 7**: Wheel_A Mode, **Table 8**: Wheel_B Mode, **Table 9**: Wheel_C Mode, **Table 10**: Wheel Avr Mode, **Table 11**: Angle, Amp Mode, **Table 12**: Centre Button Mode):

Table 7 Signals Displayed for Wheel_A Mode

	Signal 1	Signal 2	Signal 3
Description	Wheel_A Current Pad Average	Wheel_A Untouched Pad Average Level	None
Colour	Green	Pink	Yellow

Table 8 Signals Displayed for Wheel_B Mode

	Signal 1	Signal 2	Signal 3
Description	Wheel_B Current Pad Average	Wheel_B Untouched Pad Average Level	None
Colour	Green	Pink	Yellow

Table 9 Signals Displayed for Wheel_C Mode

	Signal 1	Signal 2	Signal 3
Description	Wheel_C Current Pad Average	Wheel_C Untouched Pad Average Level	None
Colour	Green	Pink	Yellow

Table 10 Signals Displayed for Wheel Avg Mode

	Signal 1	Signal 2	Signal 3
Description	Wheel_A Current Pad Average	Wheel_B Current Pad Average	Wheel_C Current Pad Average
Colour	Green	Pink	Yellow

Table 11 Signals Displayed for Angle, Amp Mode

	Signal 1	Signal 2	Signal 3
Description	Wheel Amplitude	Wheel Angle	None
Colour	Green	Pink	Yellow

Table 12 Signals Displayed for Centre Button Mode

	Signal 1	Signal 2	Signal 3
Description	Pad Total_TSCTR * $2^{DIVISORN}$	Pad Average	None
Colour	Green	Pink	Yellow

Appendix - Schematics and Layout

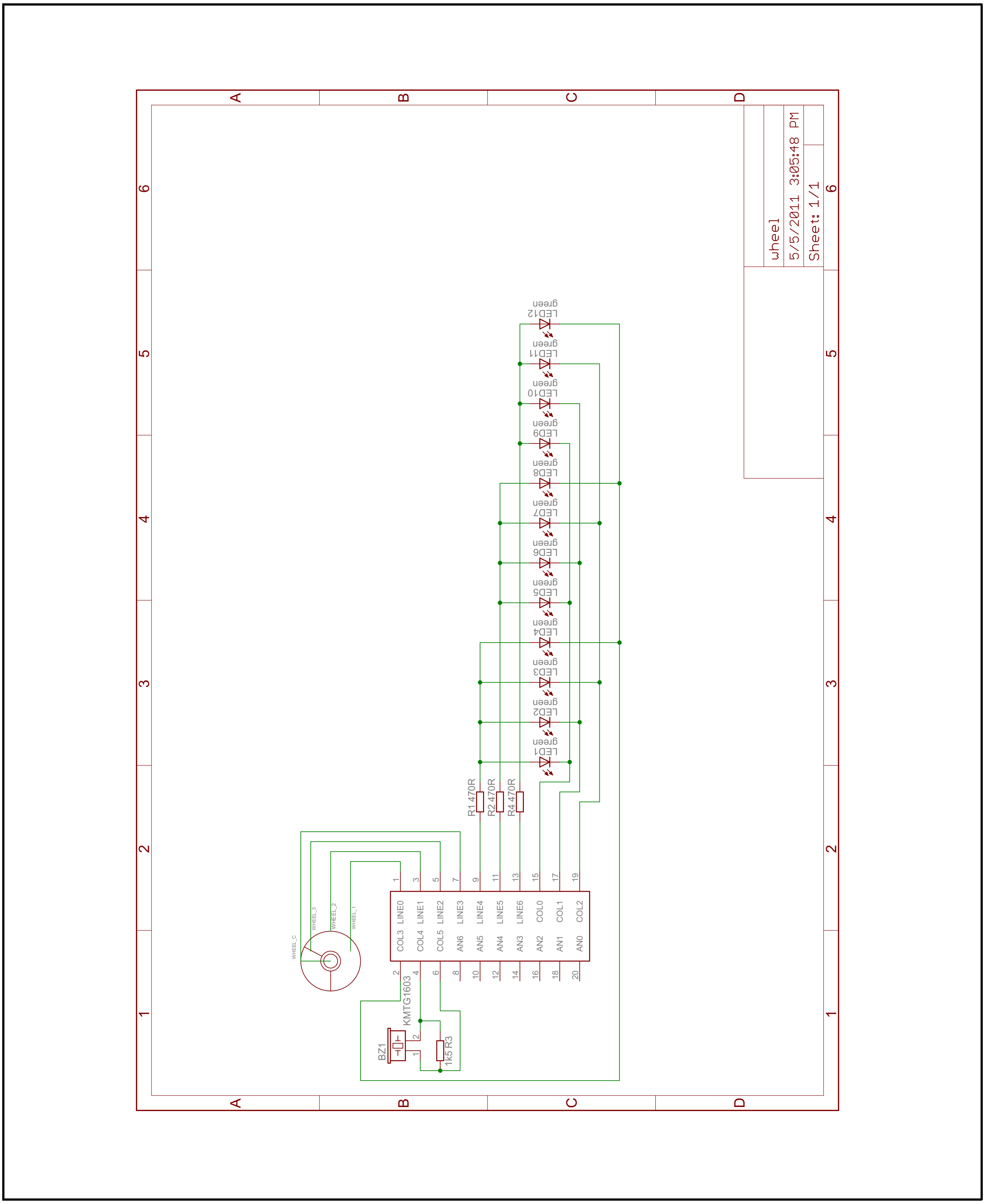

Figure 21 ***inTouch Wheel*** **Board Schematics**

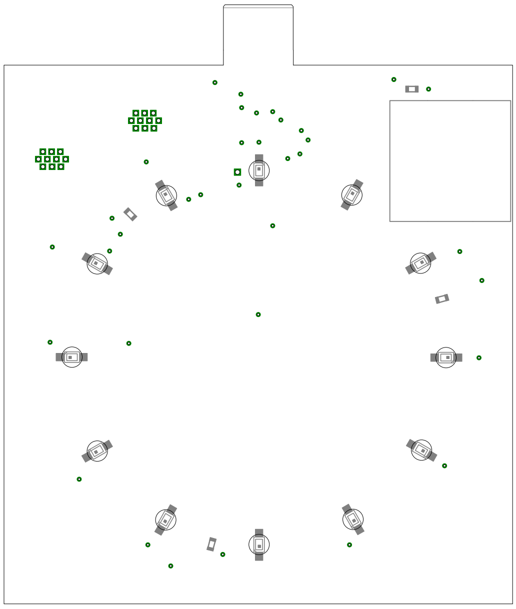

Figure 22 *inTouch Wheel* Board Component Bottom Layout

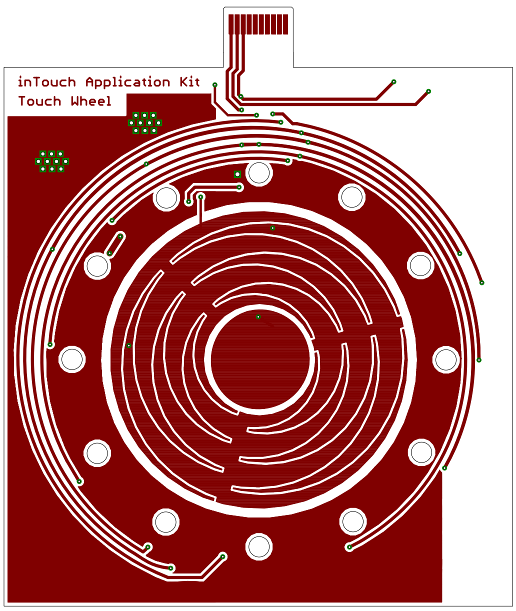

Figure 23 *inTouch Wheel* Board Top Layout

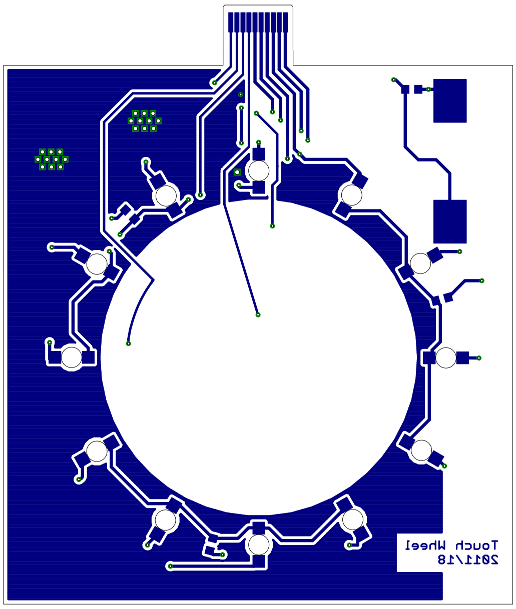

Figure 24 ***inTouch Wheel*** **Board Bottom Layout**

References

The list below provides resources that may be useful to the user.

1. User’s Manual - *XC83x; 8-Bit Single-Chip Microcontroller*
2. Application Note - *AP08100* - *Configuration for Capacitive Touch-Sense Application*
3. Application Note - *AP08110* - *Design Guidelines for XC82x and XC83x Board Layout*
4. Application Note - *AP08113* - *Capacitive-Touch Color Wheel Implementation*
5. Application Note - *AP08115* - *Design Guidelines for Capacitive Touch-Sensing Application*
6. Application Note - *AP08121* - *Infrared Remote Controller with Capacitive Touch Interface*
7. Application Note - *AP08122* - *16-Button Capacitive Touch Interface with XC836T*
8. Application Note - *AP08124* - *XC82/83x Design Guidelines for Electrical Fast Transient (EFT) Protection in Touch-Sense Applications*
9. Application Note - *AP08126* - *Infineon Touch Solutions - inTouch Application Kit*
10. Application Note - *AP08127* - *inTouch Application Kit - Buttons*
11. Application Note - *AP08129* - *inTouch Application Kit - Touch Sliders*
12. Application Note - *AP08130* - *inTouch Application Kit - LED Matrix*
13. Link to *XC83x-Series* - **www.infineon.com/xc83x**
14. Link to *Solutions for advanced touch control* - **www.infineon.com/intouch**

w w w . i n f i n e o n . c o m

Published by Infineon Technologies AG